Cairo University
Faculty of Engineering
Computer Eng. Dept.
CMPS103

Data Structures and Algorithms
LAB #1
Templates

Objectives

After this lab, the student should be able to:

- Define and use function template.
- Differentiate between function template and function overloading.
- Define and use class template.
- Create class template specialization

What is a Template?

- A template is a technique for code reuse.
- It enables the **same code logic** to be executed on **different data types**.

Types of templates

There are two types of templates:

- Function Template.
- Class Template.

- Examples of templates include:
  - A function that sorts an array of "any" data type. (Function Template)
  - A list that is required to store "any" data type. (Class Template)

Why using Template

- If you have the same code logic to be used for different data types, you can use templates instead of rewriting the same code for each new type.
- This would save code writing and maintenance time.

How to use Template

- First declare the template with **generic** data type.

```
template <typename myType>
myType getMax(myType  x, myType  y, myType  z) { ………… }
```

- Then specify that type when using the template.

```
int a,b,c,d;
……………………………………
d = getMax(a, b, c);  //Here we specify that template should operate on int vars
```

- See Code examples for more details

To Do

- Open “Examples” solution (Examples.sln).
- Run the examples in the order mentioned in the “Lab Outline” section below
- **Don’t forget to activate any example before running it.**

- For each example, make sure to read the comments, answer the questions inside it and ask about answers you're not sure about.

Lab Outline

1- Function Template:

- ❑ Function template versus function overloading
- ❑ Defining a function template (GetMax) that takes different types of arguments
- ❑ A call statement to the function (e.g. GetMax(m, n, k)) makes the compiler generate a version of the function by replacing generic type T with type of arguments m, n, and k.
- ❑ Function template can be overloaded by another function ➔ After 1st run, uncomment function **char* GetMax(char* a, char* b, char* c)**
- ❑ **Think**: What is the difference between function overloading/overriding/template?

🗐 **See Code Examples: 1_GetMax**

2- Class Template:

- ❑ Using class templates to store any data type in a class object
- ❑ How to define functions of a class template outside the class prototype
- ❑ **Limitation**: Class template should be declared in ONE .h file

☞ How to instantiate a class template to create a template class and an object

Example: **MyList<int> V1;**
MyList<T> is the ***class template (****the generic template****)***
MyList<int> is a ***template class*** (*a version of MyList for int data type*)
V1 is an **object** of class MyList<int>

🗐 **See Code Examples: 2_MyList**

3- Class Template Specialization/Customization:

As mentioned above, when instantiating an object of a class template, you should specify the type that your template should be specialized to. Assume you have a user-defined class called Car with members price and year. To create an object of class MyList (from code example #2) that is specialized for class Car, you would write:

MyList<Car> CList;

The compiler then replaces each "T" with "Car". This would lead to one of the following four cases:

Case 0: Everything is OK

All code inside class MyList<Car> compiles and works correctly for class Car with no errors.

☑ You don't need to modify anything in this case and you use code directly.

Case 1: Problem with operators

The generic template has some operators that are not defined to operate on "Car" operands

Example:
MyList member function ***getIndex*** is written as:

```
int getIndex(T Item)       //returns the index of a given Item
{
      for(int i=0; i<count; i++)
            if (data[i] == Item)
                  return i;
      return -1;    //Item no found
}
```

To check for the searched item, this function uses **==** operator to check the passed "**Item**" against items stored in the list "**data[i]**".
If the list stores objects of class Car the operator == will give an error as it cannot operate on operands of type Car.

☑ Solution: Operator Overloading

Make operator == work on Car objects through operator overloading.

🗐 **See Code Examples: 3_1_TemplateSpecialition_Op_Overloading**

Case 2: More functionality needed

MyList<Car> code has no problems but you need to update behavior of some member functions or add more functionality to the template version that is specialized for class Car.

☑ Solution: Inheritance then overriding or extension

Derive a new class from MyList<Car> (say CarList) and override/add functions to operate properly for Car objects

🗐 **See Code Examples: 3_2_TemplateSpecialition_Inheritance**

Case 3: Too many updates needed

Most of the class template functions will not work properly for certain type.
Example: MyList<char*>

☑ Solution: Class Template Specialization:

➔ create a version of MyList specifically for char* type and re-write all its function again to work properly for char*

🗐 **See Code Examples: 3_3_TemplateSpecialition_Spec**

Practice Exercises

Exercise 1

Write a function template that returns the minimum element in an array. The arguments of the function should be the array name and its size.
In main(), exercise the function with arrays of type int, long, double, char, and string.

Exercise 2

Write a class template ***Matrix*** to represent a 5x7 matrix that can store int, double, or string data types.
Provide the following member functions:

- AddValue(row, col, Value): that adds a new value given the row, column, and the value to be added
- bool BelongTo(Value): checks whether a given value belongs to the matrix or not
- PrintRow(row): Prints values in a given row
- Print( ):Prints all values in the matrix
- MaxValue: returns the maximum value in the Matrix

Exercise 3

What are the required updates to make class Matrix work for objects of class Date(with members day, month, year)? Write code to solve this problem using:

- Class Template Specialization
- Inheritance
- Operator Overloading

Which solution is the best for this case and why?

Exercise 4 (Comprehensive)

1. Class template "myPair"
    a. Create class template myPair to represent any generic key-value pair where each value is associated with certain unique key.
    Note: Both key and value members can range from simple primitive data type to any complicated user-defined data type
    b. Provide the following member functions:
        i. Non-default constructor for initialization
        ii. setPair(…..) that sets both the key and the value of the pair
        iii. Getters for key and value members separately
        iv. Overload the == operator to check whether two pairs have the same key-value combination or not

2. Use class **myPair** to create a class template "**myMap**" that has a list of generic key-value pairs.
    a. The size of the list is 100.
    b. Add the following member functions to class myMap. (For each function, decide parameters and return type).
        i. Constructor for initialization
        ii. **addPair**(myPair<……..>) ➔ Adds a new pair to the list of pairs
        iii. **getValue**(…..) ➔ Gets a value given its associated key. Returns false if the key is not found.
        iv. **count**(…..) ➔ Returns number of list pairs
        v. **updateValue**(…..) ➔ Updates a value associated with a given key. Return false if key is not found.
        vi. **displayMap**(…..) ➔ Prints all pairs in the map in form (Key, Value)
        vii. **deletePair**(myPair<…..>) ➔ Deletes a given pair if found.

3. Write a program that declares an object of class **myMap** with key of type integer and value of type string. This simple map is used to map an ID to a name.
    a. The program should display a menu for the user with the following options
        i. Add new pairs
        ii. Update existing pair
        iii. Search for a value given a key
        iv. Delete a certain pair
        v. Print map contents
    b. The user should select an operation and the program should interact with the user to collect required data and then call necessary member functions.
4. Assume you are developing a game and you have class ***Position*** that has x and y coordinates and a class hierarchy that has ***Creature*** as base class and Monster, Dragon, Solider and Player as derived classes.
    a. Show how to declare an object of class **myMap** to map each creature to a position in the game.
    b. What are the necessary updates for the above classes for myPair/myMap member functions to operate on them?